

MM	MM	000000000	MM	MM
MM	MM	000000000	MM	MM
MM	MM	000000000	MM	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000	000	MM
MM	MM	000000000	MM	MM
MM	MM	000000000	MM	MM
MM	MM	000000000	MM	MM

The diagram consists of a 10x10 grid of 100 cells. The cells are filled with the letters 'L', 'I', or 'S' in a specific pattern. The pattern is as follows:

- The central column of 10 cells (the 5th column from the left and the 5th row from the top) is filled with the letter 'I'.
- The cells in the 4th and 6th columns from the left, and the 4th and 6th rows from the top, are filled with the letter 'S'.
- All other cells are filled with the letter 'L'.

For example, the cell at the 5th column and 5th row contains an 'I', while the cell at the 3rd column and 3rd row contains an 'S'. The entire pattern is centered within the 10x10 grid.

```
1 0001 0
2 0002 0 XTITLE 'Maintenance Operations NPARSE action routines for parsing parameters'
3 0003 0 MODULE MOMPARSE (
4 0004 0           LANGUAGE (BLISS32),
5 0005 0           ADDRESSING_MODE (NONEXTERNAL=GENERAL),
6 0006 0           ADDRESSING_MODE (EXTERNAL=GENERAL),
7 0007 0           IDENT = 'V04-000'
8 0008 0           ) =
9 0009 1 BEGIN
10 0010 1 ****
11 0011 1 *
12 0012 1 *
13 0013 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
14 0014 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
15 0015 1 * ALL RIGHTS RESERVED.
16 0016 1 *
17 0017 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
18 0018 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
19 0019 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
20 0020 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
21 0021 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
22 0022 1 * TRANSFERRED.
23 0023 1 *
24 0024 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
25 0025 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
26 0026 1 * CORPORATION.
27 0027 1 *
28 0028 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
29 0029 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
30 0030 1 *
31 0031 1 *
32 0032 1 ****
33 0033 1 *
34 0034 1 *
35 0035 1 ++
36 0036 1 * FACILITY: DECnet-VAX V2.0 Network Management Listener
37 0037 1 *
38 0038 1 *
39 0039 1 * ABSTRACT:
40 0040 1 * This module contains action routines called by NPARSE to parse and
41 0041 1 * store NICE entity parameters.
42 0042 1 *
43 0043 1 * ENVIRONMENT: VAX/VMS Operating System
44 0044 1 *
45 0045 1 * AUTHOR: Kathy Perko
46 0046 1 *
47 0047 1 * CREATION DATE: 2-Jan-1983
48 0048 1 *
49 0049 1 * MODIFIED BY:
50 0050 1 *     V03-005 MKP0005      Kathy Perko      13-July-1984
51 0051 1 *           Change NODE SERVICE PASSWORD from an H-8 field to an HI-8
52 0052 1 *           field. The architecture conflicts with itself about it.
53 0053 1 *
54 0054 1 *     V03-004 MKP0004      Kathy Perko      6-June-1984
55 0055 1 *           Don't apply area 1 fix to exec.
56 0056 1 *
57 0057 1 *     V03-003 MKP0003      Kathy Perko      1-May-1984
```

: 58 0058 1 : Check for correct loop assistant parameter ID in LOOP
: 59 0059 1 : CIRCUIT parameter consistency check.
: 60 0060 1 :
: 61 0061 1 : V03-002 MKP0002 Kathy Perko 28-Mar-1984
: 62 0062 1 : Fix area 1 problem.
: 63 0063 1 :
: 64 0064 1 : V03-001 MKP0001 Kathy Perko 29-Jan-1984
: 65 0065 1 : Do some cross checking on LOOP CIRCUIT parameters.
: 66 0066 1 : Add a routine to check for loopback assist request messages,
: 67 0067 1 : and a routine to use the MOP message software ID field as
: 68 0068 1 : a load file ID.
: 69 0069 1 :
: 70 0070 1 !--
: 71 0071 1 :

```
73 0072 1 %SBTTL 'Declarations'  
74 0073 1  
75 0074 1 !  
76 0075 1 ! TABLE OF CONTENTS:  
77 0076 1 !  
78 0077 1 !  
79 0078 1 FORWARD ROUTINE  
80 0079 1     momSparse_nice_entity,  
81 0080 1     momSparse_function,  
82 0081 1     momSparse_option,  
83 0082 1     momSparse_entity_id,  
84 0083 1     mom$save_param,  
85 0084 1     mom$save_node_id,  
86 0085 1     mom$check_node_entity,  
87 0086 1     mom$check_loop_params,  
88 0087 1     mom$mob_chk_loop_assist,  
89 0088 1     mom$save_mop_msg,  
90 0089 1     mom$save_load_file_id,  
91 0090 1     mom_fix_node_num: NOVALUE,  
92 0091 1     momSparse_error: NOVALUE,  
93 0092 1     mom$prsmoperr;  
94 0093 1  
95 0094 1 !  
96 0095 1 ! INCLUDE FILES:  
97 0096 1 !  
98 0097 1 !  
99 0098 1 LIBRARY 'LIB$:MOMLIB.L32';  
100 0099 1 LIBRARY 'SHRLIB$:NMLIBRY.L32';  
101 0100 1 LIBRARY 'SHRLIB$:NET.L32';  
102 0101 1 LIBRARY 'SYS$LIBRARY:STARLET.L32';  
103 0102 1 !  
104 0103 1 !  
105 0104 1 ! EXTERNAL REFERENCES:  
106 0105 1 !  
107 0106 1 !  
108 0107 1 $MOM_externals;           ! Macro with common MOM externals.  
109 0108 1  
110 0109 1 EXTERNAL_LITERAL  
111 0110 1     mom$badmopfct;  
112 0111 1  
113 0112 1 EXTERNAL  
114 0113 1     mom$npa_init,  
115 0114 1     mom$ab_ncp_version;    ! Nparse table for NICE message entities.  
116 0115 1  
117 0116 1 EXTERNAL ROUTINE  
118 0117 1     mom$nparse,  
119 0118 1     mom$build_p2,  
120 0119 1     mom$netacp_qio,  
121 0120 1     mom$error;
```

```

: 123 0121 1 %SBTTL 'momSparse_nice_entity Initial message parsing routine'
: 124 0122 1 GLOBAL ROUTINE momSparse_nice_entity =
: 125 0123 1
: 126 0124 1 ++
: 127 0125 1 FUNCTIONAL DESCRIPTION:
: 128 0126 1 This routine invokes the NPARSE facility to check the function,
: 129 0127 1 option, and entity codes in a NICE request received from NCP.
: 130 0128 1
: 131 0129 1 IMPLICIT OUTPUTS:
: 132 0130 1
: 133 0131 1 MOMSGB_FUNCTION contains the function code.
: 134 0132 1 MOMSGB_OPTION_BYTE contains the option codes.
: 135 0133 1 MOMSGL_ENTITY_CODE contains the entity code.
: 136 0134 1 MOMSAB_NPARSE_BLK contains parsing information about the remainder
: 137 0135 1 of the message.
: 138 0136 1
: 139 0137 1 ROUTINE VALUE:
: 140 0138 1 COMPLETION CODES:
: 141 0139 1 If the parse fails then the error is signalled, and a NICE error
: 142 0140 1 response is built with the error specified by the parse state
: 143 0141 1 table. Otherwise success is returned.
: 144 0142 1
: 145 0143 1 --
: 146 0144 1
: 147 0145 2 BEGIN
: 148 0146 2
: 149 0147 2 LOCAL
: 150 0148 2   status;           ! Temporary status
: 151 0149 2
: 152 0150 2 Initialize message parsing data
: 153 0151 2
: 154 0152 2 momSgl_service_flags = 0;           ! Clear internal options flags
: 155 0153 2
: 156 0154 2 Initialize the NPARSE argument block with the address and length
: 157 0155 2 of the NICE message to be parsed. Then call the NPARSE facility
: 158 0156 2 to parse the function, option, and entity fields of the message.
: 159 0157 2
: 160 0158 2 momSab_nparse_blk [npasl_msgptr] = momSab_nice_rcv_buf;
: 161 0159 2 momSab_nparse_blk [npasl_msgcnt] = .momSgl_nice_rcv_msg_len;
: 162 0160 2
: 163 0161 2 nma$npars (momSab_nparse_blk, momSnpa_init);
: 164 0162 2
: 165 0163 2 If control returns here, the message parsed correctly. Otherwise,
: 166 0164 2 an error was signalled and an error response returned to NCP via
: 167 0165 2 NML.
: 168 0166 2
: 169 0167 2 RETURN SUCCESS
: 170 0168 2
: 171 0169 1 END;           ! End of MOMPARE_NICE_ENTITY

```

```

:TITLE MOMPARE Maintenance Operations NPARSE action r
:        outines f
:IDENT \V04-000\

:EXTRN MOMSGL_LOGMASK, MOMSGL_SVD_INDEX
:EXTRN MOMSAB_SERVICE_DATA

```

.EXTRN MOMSGB_FUNCTION
.EXTRN MOMSGB_OPTION_BYTE
.EXTRN MOMSGB_ENTITY_CODE
.EXTRN MOMSAB_ENTITY_BUF
.EXTRN MOMSGQ_ENTITY_BUF_DSC
.EXTRN MOMSGL_SERVICE_FLAGS
.EXTRN MOMSAB_NPARSE_BLK
.EXTRN MOMSAB_NICE_RCV_BUF
.EXTRN MOMSAB_NICE_XMIT_BUF
.EXTRN MOMSGQ_NICE_RCV_BUF_DSC
.EXTRN MOMSGL_NICE_RCV_MSG_LEN
.EXTRN MOMSGQ_NICE_XMIT_BUF_DSC
.EXTRN MOMSAB_MSGBLOCK
.EXTRN MOMSAB_ACPQIO_BUFFER
.EXTRN MOMSGQ_ACPQIO_BUF_DSC
.EXTRN MOMSAB_CIB, MOMSAB_LOOP_CIB
.EXTRN MOMSAB_TRIGGER_CIB
.EXTRN MOMSAB_MOP_XMIT_BUF
.EXTRN MOMSGQ_MOP_XMIT_BUF_DSC
.EXTRN MOMSAB_MOP_RCV_BUF
.EXTRN MOMSGQ_MOP_RCV_BUF_DSC
.EXTRN MOMSAB_MOP_MSG, MOMSGQ_MOP_MSG_DSC
.EXTRN MOMSGW_EVT_CODE
.EXTRN MOMSGB_EVT_POPR
.EXTRN MOMSGB_EVT_PRSN
.EXTRN MOMSGB_EVT_PSER
.EXTRN SVD\$GK_PCNO_ADD
.EXTRN SVD\$GK_PCNO_SDV
.EXTRN SVD\$GK_PCNO_CPU
.EXTRN SVD\$GK_PCNO_STY
.EXTRN SVD\$GK_PCNO_DAD
.EXTRN SVD\$GK_PCNO_DCT
.EXTRN SVD\$GK_PCNO_IHO
.EXTRN SVD\$GK_PCNO_NNA
.EXTRN SVD\$GK_PCNO_SLI
.EXTRN SVD\$GK_PCNO_SPA
.EXTRN SVD\$GK_PCNO_HWA
.EXTRN SVD\$GK_PCNO_SNV
.EXTRN SVD\$GK_PCNO_LOA
.EXTRN SVD\$GK_PCNO_SLO
.EXTRN SVD\$GK_PCNO_TLO
.EXTRN SVD\$GK_PCNO_DFL
.EXTRN SVD\$GK_PCNO_SID
.EXTRN SVD\$GK_PCNO_DUM
.EXTRN SVD\$GK_PCNO_SDU
.EXTRN SVD\$GK_PCNO_SHNA
.EXTRN SVD\$GK_PCNO_SHHW
.EXTRN SVD\$GK_PCNO_SFTY
.EXTRN SVD\$GK_PCNO_PHA
.EXTRN SVD\$GK_PCNO_SDA
.EXTRN SVD\$GK_PCNO_LPC
.EXTRN SVD\$GK_PCNO_LPL
.EXTRN SVD\$GK_PCNO_LPD
.EXTRN SVD\$GK_PCNO_LPH
.EXTRN SVD\$GK_PCNO_LPA
.EXTRN SVD\$GK_PCNO_LPN
.EXTRN SVD\$GK_PCNO_SLNA

```

        .EXTRN  SVDSGK$PCNO$LNH
        .EXTRN  SVDSGK$PCNO$LAN
        .EXTRN  SVDSGK$PCNO$LN
        .EXTRN  SVDSGK$PCNO$LAH
        .EXTRN  SVDSGK$PCLI$TI
        .EXTRN  SVDSC$ENTRY$COUNT
        .EXTRN  MOMS$BADMOPFCT, MOMSNPA$INIT
        .EXTRN  MOMS$AB$NCP$VERSION
        .EXTRN  NMASNP$PARSE, MOMSBUILD$P2
        .EXTRN  MOMSNETACP$QIO, MOM$ERROR

        .PSECT  $CODE$, NOWRT, 2

        52 00000000G 00 9E 00002 0004 00000 .ENTRY  MOMSPARSE$NICE$ENTITY, Save R2 : 0122
        00000000G 00 D4 00009                   MOVAB  MOMS$AB$NP$PARSE$BLK+8, R2
        62 00000000G 00 9E 0000F                   CLRL  MOM$GL$SERVICE$FLAGS : 0152
        FC  A2 00000000G 00 D0 00016                   MOVAB  MCM$AB$NICE$RCV$BUF, MOM$AB$NP$PARSE$BLK+8 : 0158
        00000000G 00 9F 0001E                   MOVL  MOM$GL$NICE$RCV$MSG$LEN, - : 0159
        F8  A2 9F 00024                   PUSHAB MOMSNPA$INIT : 0161
        00000000G 00 02 FB 00027                   PUSHAB MOMS$AB$NP$PARSE$BLK
        50  01 D0 0002E                   CALLS #2, NMASNP$PARSE
        04 00031                   MOVL  #1, R0 : 0167
                                         RET : 0169

```

: Routine Size: 50 bytes. Routine Base: \$CODE\$ + 0000

Maintenance Operations NPARSE action routines f 16-Sep-1984 02:06:08
mom\$parse_function Store function code (action 14-Sep-1984 12:44:36) 6
VAX-11 Bliss-32 V4.0-742 Page 7
DISK\$VMSMASTER:[MOM.SRC]MOPARSE.B32;1 (4)

```
0170 1 %SBTTL 'mom$parse_function Store function code (action routine)'  
0171 1 GLOBAL ROUTINE mom$parse_function =  
0172 1 !++  
0173 1 FUNCTIONAL DESCRIPTION:  
0174 1  
0175 1 Parse and store the function code from the NICE command message.  
0176 1  
0177 1 IMPLICIT OUTPUTS:  
0178 1  
0179 1 MOM$GB_FUNCTION contains the function code.  
0180 1  
0181 1 ROUTINE VALUE:  
0182 1 COMPLETION CODES:  
0183 1  
0184 1 Always returns success (SSS_NORMAL).  
0185 1  
0186 1  
0187 1 --  
0188 1  
0189 2 BEGIN  
0190 2  
0191 2 $npa_argdef; ! Define NPARSE block reference  
0192 2  
0193 2 mom$gb_funcatio = .nparse_block [npa$b_byte]; ! Set function  
0194 2  
0195 2 RETURN sss_normal  
0196 2  
0197 1 END; ! End of MOM$PARSE_FUNCTION
```

00000000G	00	18	0000 00000	.ENTRY	MOM\$PARSE_FUNCTION, Save nothing	0171
	50		AC 90 00002	MOVB	24(NPARSE_BLOCK), MOM\$GB_FUNCTION	0193
			01 D0 0000A	MOVL	#1, R0	0195
			04 0000D	RET		0197

; Routine Size: 14 bytes, Routine Base: \$CODE\$ + 0032

M 6

Maintenance Operations NPARSE action routines f 16-Sep-1984 02:06:08
momSparse_option Store NICE message option byt 14-Sep-1984 12:44:36

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOM.SRC]MOMPARSE.B32;1 (5)

Page 8
MON
V04

```

: 202 0198 1 %SBTTL 'momSparse_option Store NICE message option byte (action routine)'
: 203 0199 1 GLOBAL ROUTINE momSparse_option =
: 204 0200 1
: 205 0201 1 !++
: 206 0202 1 !++ FUNCTIONAL DESCRIPTION:
: 207 0203 1 This routine is a NPARSE action routine that is called while
: 208 0204 1 parsing a NICE message. It saves the option byte in a global
: 209 0205 1 field.
: 210 0206 1
: 211 0207 1 !++ IMPLICIT INPUTS:
: 212 0208 1 NPARSE_BLOCK [NPASB_BYTE] contains the option byte.
: 213 0209 1
: 214 0210 1 !++ ROUTINE VALUE:
: 215 0211 1 !++ COMPLETION CODES:
: 216 0212 1 Success (SSS_NORMAL) is always returned.
: 217 0213 1
: 218 0214 1 !--
: 219 0215 1
: 220 0216 2 BEGIN
: 221 0217 2
: 222 0218 2 $npa_argdef; ! Define NPARSE block reference
: 223 0219 2
: 224 0220 2 ! Save the entity code from the NPARSE argument block
: 225 0221 2
: 226 0222 2 mom$gb_option_byte = .nparse_block [npasb_byte];
: 227 0223 2
: 228 0224 2 RETURN sss_normal
: 229 0225 2
: 230 0226 1 END; ! End of MOMPARSE_OPTION

```

00000000G	00	18	0000 0000	.ENTRY	MOMPARSE_OPTION, Save nothing	: 0199
	50		AC 90 00002	MOVB	24(NPARSE_BLOCK), MOM\$GB_OPTION_BYTE	: 0222
			01 D0 0000A	MOVL	#1, R0	: 0224
			04 0000D	RET		: 0226

; Routine Size: 14 bytes. Routine Base: \$CODE\$ + 0040

```
0227 1 %SBTTL 'mom$parse_entity_id Parse the service id'
0228 1 GLOBAL ROUTINE mom$parse_entity_id =
0229 1
0230 1 !++
0231 1 !++
0232 1 !+ FUNCTIONAL DESCRIPTION:
0233 1 !+ Parse the service id code from the MOP message or NICE command.
0234 1
0235 1 !+ IMPLICIT INPUTS:
0236 1 !+ NPARSE_BLOCK [NPASL PARAM] contains the MOM internal entity code
0237 1 !+ (MOM$C_CIRCUIT, MOM$C_LINE, MOM$C_NODE, or MOM$C_NODEBYNAME).
0238 1
0239 1 !+ OUTPUTS:
0240 1 !+ MOM$AB_ENTITY_BUF contains the entity ID
0241 1 !+ MOM$CQ_ENTITY_BUF_DSC contains a descriptor of the entity ID in
0242 1 !+ MOM$AB_ENTITY_BUF.
0243 1 !+ MOM$GB_ENTITY_CODE contains the MOM internal code for the entity.
0244 1 !--
0245 2 BEGIN
0246 2
0247 2 $npa_argdef;
0248 2
0249 2 LOCAL
0250 2   adr,
0251 2   ent,
0252 2   len,
0253 2   svd_index;
0254 2
0255 2 ent = .nparse_block [npasl_param];
0256 2
0257 2 ! Select parse table according to entity code.
0258 2
0259 2 SELECTU .ent OF
0260 2   SET
0261 2
0262 2   [mom$C_node]:
0263 3     BEGIN
0264 3     MAP
0265 3       adr: REF BBLOCK;
0266 3       len = 2;
0267 3       adr = .nparse_block [npasl_fldptr];
0268 3       svd_index = svd$gk_pcno_add;
0269 3
0270 3       ! If the node area is 0 and it's not a Phase III (or less) NCP,
0271 3       ! change to area 1.
0272 3
0273 3       mom_fix_node_num (.adr);
0274 2
0275 2   END;
0276 2
0277 2   [mom$C_line, mom$C_nodebyname]:
0278 3     BEGIN
0279 3       len = .(.nparse_block [npasl_fldptr])<0,8>;
0280 3       adr = .nparse_block [npasl_fldptr] + 1;
0281 3       IF .ent EQL mom$C_line THEN
0282 3         svd_index = svd$gk_pcno_sli
0283 3       ELSE
0284 3         svd_index = svd$gk_pcno_nna;
```

```

289 0284 2      END;
290 0285 2
291 0286 2      [mom$sc_circuit]:
292 0287 3      BEGIN
293 0288 3          len = .(nparse_block [npasl_fldptr])<0,8>;
294 0289 3          adr = .nparse_b[lock [npasl_fldptr] + 1;
295 0290 3          svd_index = svd$gk_pcno_sli;
296 0291 2      END;
297 0292 2
298 0293 2      [ALWAYS]:
299 0294 3      BEGIN
300 0295 3          CH$MOVE (.len, .adr, mom$ab_entity_buf);
301 0296 3          mom$gq_entity_buf_dsc [0] = .len;
302 0297 3
303 0298 3          | Put the entity ID into the Service Data Table so it will
304 0299 3          | override the value returned from the volatile database.
305 0300 3
306 0301 3          mom$ab_service_data [.svd_index, svd$b_string_len] = .len;
307 0302 3          CH$MOVE (.len,
308 0303 3              .adr,
309 0304 3                  mom$ab_service_data [.svd_index, svd$t_string]);
310 0305 3          mom$ab_service_data [.svd_index, svd$v_msg_param] = true;
311 0306 2      END;
312 0307 2
313 0308 2      TES:
314 0309 2
315 0310 2      ! Save the entity code.
316 0311 2
317 0312 2      mom$gb_entity_code = .ent;
318 0313 2
319 0314 2      RETURN ss$normal
320 0315 2
321 0316 1 END;

```

! End of mom\$parse_entity_id

			0FFC 00000		.ENTRY	MOMPARSE_ENTITY_ID, Save R2,R3,R4,R5,R6,- ; 0228
			5B 00000000G	8F D0 00002	MOVL	#SVDS\$GK PCNO SLI, R11
			5A 00000000G	00 9E 00009	MOVAB	MOM\$AB SERVICE DATA+8, R10
			59 20	AC D0 00010	MOVL	32(NPARSE_BLOCK), ENT
				17 12 00014	BNEQ	1\$
			57	02 D0 00016	MOVL	#2, LEN
			58 14	AC D0 00019	MOVL	20(NPARSE_BLOCK), ADR
			56 00000000G	8F D0 0001D	MOVL	#SVDS\$GK_PCNO_ADD, SVD_INDEX
				58 DD 00024	PUSHL	ADR
			00000000V	00 01 FB 00026	CALLS	#1, MOM_FIX_NODE_NUM
				01 59 D1 0002D	CMPL	ENT, #1
				1\$:	BEQL	2\$
				05 13 00030	CMPL	ENT, #3
				03 59 D1 00032	BNEQ	4\$
				1A 12 00035	MOVZBL	@20(NPARSE_BLOCK), LEN
			58 14	57 14 BC 9A 00037	ADDL3	#1, 20(NPARSE_BLOCK), ADR
				03 01 C1 00038	CMPL	ENT, #3
				05 59 D1 00040	BNEQ	3\$
				05 12 00043		

		56	5B	D0	00045	MOVL	R11, SVD_INDEX	: 0281	
			07	11	00048	BRB	4\$: 0283	
		56 00000000G	8F	D0	0004A	3\$:	MOVL #SVD\$GK_PCNO_NNA, SVD_INDEX	: 0286	
		02	59	D1	00051	4\$:	CMPL ENT, #2	: 0288	
			0C	12	00054	BNEQ	5\$: 0289	
	58	14	57	14	00056	MOVZBL	@20(NPARSE_BLOCK), LEN	: 0290	
00000000G	00		AC	01	C1	0005A	ADDL3 #1, 20(NPARSE_BLOCK), ADR	: 0295	
			56	5B	D0	0005F	MOVL R11, SVD_INDEX	: 0296	
		00000000G	68	57	28	00062	5\$:	MOVC3 LEN, (ADR), MOM\$AB_ENTITY_BUF	: 0301
		00	00	57	D0	0006A	MOVL LEN, MOM\$GQ_ENTITY_BUF_DSC	: 0304	
			56 00000089	8F	C4	00071	MULL2 #137, R6	: 0305	
01 AA46		6A46		57	90	00078	MOVB LEN, MOM\$AB_SERVICE_DATA+8[R6]	: 0312	
			68	57	28	0007C	MOVC3 LEN, (ADR), MOM\$AB_SERVICE_DATA+9[R6]	: 0314	
		FF AA46		01	88	00082	BISB2 #1, MOM\$AB_SERVICE_DATA+7[R6]	: 0316	
		00000000G	00	59	90	00087	MOVB ENT, MOM\$GB_ENTITY_CODE		
			50	01	D0	0008E	MOVL #1, R0		
				04	00091	RET			

: Routine Size: 146 bytes, Routine Base: \$CODE\$ + 004E

: 322 0317 1

```
324 0318 1 %SBTTL 'mom$save_param Save NICE parameter value'  
325 0319 1 GLOBAL ROUTINE mom$save_param =  
326 0320 1  
327 0321 1 !++  
328 0322 1 : FUNCTIONAL DESCRIPTION:  
329 0323 1 : This is an NPARSE action routine that is called while parsing  
330 0324 1 : a NICE message from NCP or a MOP message from the target node.  
331 0325 1 : It saves a parameter in the Service Data Table and sets a flag  
332 0326 1 : to indicate that the parameter from the volatile database is  
333 0327 1 : not to be used for this operation (since one was supplied in  
334 0328 1 : the NICE or MOP message).  
335 0329 1  
336 0330 1 : IMPLICIT INPUTS:  
337 0331 1 : NPARSE_BLOCK (pointed to by AP) contains the parsed parameter data.  
338 0332 1 : NPASL_FLDcnt is the parameter length.  
339 0333 1 : NPASL_Fldptr is a pointer to the parameter in the received  
340 0334 1 : message buffer.  
341 0335 1 : MOM$GL_SVD_INDEX contains the index into the Service Data table  
342 0336 1 : (MOM$AB_SERVICE_DATA).  
343 0337 1  
344 0338 1 : IMPLICIT OUTPUTS:  
345 0339 1 : The parameter value or string is inserted into the Service Data Table.  
346 0340 1  
347 0341 1 : ROUTINE VALUE:  
348 0342 1 : COMPLETION CODES:  
349 0343 1 : Always returns SSS_NORMAL.  
350 0344 1  
351 0345 1 --  
352 0346 1  
353 0347 2 BEGIN  
354 0348 2  
355 0349 2 $npa_argdef; ! Define NPARSE block reference  
356 0350 2  
357 0351 2 LOCAL  
358 0352 2 svd_index; ! Index into this parameter's entry in  
359 0353 2 : the Service Data table.  
360 0354 2 msgsize; ! Resultant message size  
361 0355 2 len; ! Temporary parameter pointer  
362 0356 2 ptr;  
363 0357 2  
364 0358 2 : Add descriptor entry for this parameter.  
365 0359 2  
366 0360 2 len = .nparse_block [npasl_fldcnt];  
367 0361 2 ptr = .nparse_block [npasl_fldptr];  
368 0362 2  
369 0363 2 : If the NPARSE tables specified a parameter, then it is the SVD (Service  
370 0364 2 : Data table) index. This is true only when parsing MOP messages. When  
371 0365 2 : parsing NICE messages, the SVD index must be saved when the parameter  
372 0366 2 : ID is parsed; this routine is not called until parsing reaches the  
373 0367 2 : parameter value.  
374 0368 2  
375 0369 2 IF .nparse_block [npasl_param] NEQ 0 THEN  
376 0370 2 : svd_index = .nparse_block [npasl_param]  
377 0371 2 ELSE  
378 0372 2 : svd_index = .mom$gl_svd_index;  
379 0373 2  
380 0374 2 : Save the parameter in the Service Data Table.
```

```

: 381      0375 2 !
: 382      2 IF .mom$ab_service_data [.svd_index, svd$b_nice_type]
: 383      22      EQL svd$k_string THEN
: 384      BEGIN
: 385      Len = .len - 1;
: 386      CH$MOVE (.len, (.ptr + 1), mom$ab_service_data [.svd_index, svd$t_string]);
: 387      mom$ab_service_data [.svd_index, svd$b_string_len] = .len;
: 388      mom$ab_service_data [.svd_index, svd$v_msg_param] = true;
: 389      END
: 390      2 ELSE
: 391      BEGIN
: 392      Save the parameter value.
: 393      CH$COPY (.len,
: 394      .ptr,
: 395      0,
: 396      4,
: 397      mom$ab_service_data [.svd_index, svd$l_param]);
: 398      mom$ab_service_data [.svd_index, svd$v_msg_param] = true;
: 399      END;
: 400      2 ! Clear SVD index because the parsing routines think they are simply
: 401      2 ! setting a bit when they put the index into this variable.
: 402      2 ! mom$gl_svd_index = 0;
: 403      2 RETURN ss$_normal
: 404      2
: 405      1 END;
: 406      ! End of MOM$SAVE_PARAM

```

		07FC 00000	.ENTRY	MOM\$SAVE_PARAM, Save R2,R3,R4,R5,R6,R7,R8,-	0319
		5A 00000000G	MOVAB	MOM\$GL_SVD_INDEX, R10	
		59 00000000G	MOVAB	MOM\$AB_SERVICE_DATA+9, R9	
		58 10 AC 00 00010	MOVL	16(NPARSE_BLOCK), LEN	0360
		51 14 AC 00 00014	MOVL	20(NPARSE_BLOCK), PTR	0361
		20 AC 05 00018	TSTL	32(NPARSE_BLOCK)	0369
		06 13 0001B	BEQL	1\$	
		50 20 AC 00 0001D	MOVL	32(NPARSE_BLOCK), SVD_INDEX	0370
		03 11 00021	BRB	2\$	
		50 6A 00 00023 1\$:	MOVL	MOM\$GL_SVD_INDEX, SVD_INDEX	0372
56	50	50 00000089 8F C5 00026 2\$:	MULL3	#137, SVD_INDEX, R6	0376
		56 59 C1 0002E	ADDL3	R9, R6, R0	0380
		57 FE A946 9E 00032	MOVAB	MOM\$AB_SERVICE_DATA+7[R6], R7	0382
		03 FD A946 91 00037	CMPB	MOM\$AB_SERVICE_DATA+6[R6], #3	0377
		0E 12 0003C	BNEQ	3\$	
		58 D7 0003E	DECL	LEN	0379
60	01 A1	58 28 00040	MOVC3	LEN, 1(PTR), (R0)	0380
	FF A946	58 90 00045	MOVB	LEN, MOM\$AB_SERVICE_DATA+8[R6]	0381
		06 11 0004A	BRB	4\$	0382
04	00	61 58 2C 0004C 3\$:	MOVC5	LEN, (PTR), #0, #4, (R0)	0393
		60 00051			

MOMPARSE
V04-000

Maintenance Operations NPARSE action routines f 16-Sep-1984 02:06:08 F 7
mom\$save_param Save NICE parameter value 14-Sep-1984 12:44:36 VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOM.SRC]MOMPARSE.B32;1 Page 14
(7)

67	01 88 00052 4\$:	BISB2	#1, (R7)	: 0394
	6A 04 00055	CLRL	MOMISGL_SVD_INDEX	: 0400
50	01 D0 00057	MOVL	#1, R0	: 0402
	04 0005A	RET		: 0404

: Routine Size: 91 bytes. Routine Base: \$CODE\$ + 00E0

```
; 412      0405 1 ISBTTL 'mom$save_node_id'          Save node id'
; 413      0406 1 GLOBAL ROUTINE mom$save_node_id =
; 414
; 415      0407 1 !++
; 416      0408 1 !++ FUNCTIONAL DESCRIPTION:
; 417      0409 1 This is an NPARSE action that saves a node id passed in
; 418      0410 1 a LOAD, TRIGGER, or LOOP command.
; 419
; 420      0411 1 !++ IMPLICIT INPUTS:
; 421      0412 1 NPARSE_BLOCK [NPASL_FLDPTR] contains the pointer to the entity
; 422      0413 1 format code and id string.
; 423
; 424      0414 1 !++ ROUTINE VALUE:
; 425      0415 1 COMPLETION CODES:
; 426      0416 1 Always returns success (SSS_NORMAL).
; 427
; 428      0417 1 !-- BEGIN
; 429
; 430      0418 2 BEGIN
; 431      0419 2 !-- LOCAL
; 432      0420 2 !-- $npa_argdef;           ! Define NPARSE block reference
; 433
; 434      0421 2 !-- node_addr_svd,
; 435      0422 2 !-- node_name_svd,
; 436      0423 2 !-- length,
; 437      0424 2 !-- addr;
; 438
; 439      0425 2 !-- The LOAD HOST parameter is a word rather than a node id (for which
; 440      0426 2 !-- the node address is preceded by a byte of 0).
; 441
; 442      0427 2 !-- IF .nparse_block [npasl_param] EQL mom$node_addr_param THEN
; 443      0428 3 BEGIN
; 444      0429 3 !-- length = 0;
; 445      0430 3 !-- addr = .nparse_block [npasl_fldptr];
; 446
; 447      0431 3 !-- END
; 448      0432 2 !-- ELSE
; 449      0433 3 BEGIN
; 450
; 451      0434 3 !-- Get length and address of node id string.
; 452
; 453      0435 3 !-- length = .(nparse_block [npasl_fldptr])<0,8>; ! Get length
; 454      0436 3 !-- addr = .nparse_block [npasl_fldptr] + 1;
; 455
; 456      0437 2 SELECTONEU .mom$gl_svd_index OF
; 457      0438 2 !-- SET
; 458      0439 2 !-- [svd$gk_pcno_aho]:
; 459      0440 3 BEGIN
; 460      0441 3 !-- node_addr_svd = svd$gk_pcno_aho;
; 461      0442 3 !-- node_name_svd = svd$gk_pcno_shna;
; 462
; 463      0443 2 !-- END;
; 464      0444 2 !-- [svd$gk_pcno_lpn]:
; 465      0445 3 BEGIN
; 466      0446 3 !-- node_addr_svd = svd$gk_pcno_lpn;
; 467      0447 3 !-- node_name_svd = svd$gk_pcno_slna;
; 468
; 469      0448 2 !-- END;
; 470      0449 2 !-- [svd$gk_pcno_lan]:
```

Maintenance Operations NPARSE action routines f 16-Sep-1984 02:06:08 14-Sep-1984 12:44:36 VAX-11 Bliss-32 V4.0-742 DISKSVMSMASTER:[MOM.SRC]MOPPARSE.B32;1 Page 16 (8)

```

469 0462 3      BEGIN
470 0463 3      node_addr_svd = svd$gk_pcno_lan;
471 0464 3      node_name_svd = svd$gk_pcno_slnn;
472 0465 2      END;
473 0466 2      TES;
474 0467 2      | If length is zero then id is a node address, otherwise it is a
475 0468 2      | node name string.
476 0469 2      | IF .length EQ 0 THEN
477 0470 2      |     Save the node address.
478 0471 2      |     BEGIN
479 0472 2      |     BIND
480 0473 2      |     node_addr = mom$ab_service_data [.node_addr_svd, svd$1_param] :
481 0474 2      |     BBLOCK;
482 0475 2      |     mom$ab_service_data [.node_addr_svd, svd$1_param] = (.addr)<0,16>;
483 0476 2      |     mom$ab_service_data [.node_addr_svd, svd$1v_msg_param] = true;
484 0477 2      |     If the node area is 0 and it's not a Phase III (or less) NCP,
485 0478 2      |     change to area 1.
486 0479 2      |     mom_fix_node_num (node_addr);
487 0480 2      |     END
488 0481 2      | ELSE
489 0482 2      |     If it's a node name, save it and get the node address from the
490 0483 2      |     volatile database.
491 0484 2      |     BEGIN
492 0485 2      |     mom$ab_service_data [.node_name_svd, svd$2_string_len] = .length;
493 0486 2      |     CH$MOVE (.length, .addr,
494 0487 2      |     mom$ab_service_data [.node_name_svd, svd$2t_string]);
495 0488 2      |     mom$ab_service_data [.node_name_svd, svd$2v_msg_param] = true;
496 0489 2      |     END;
497 0490 2      |     mom$gl_svd_index = 0;           ! Reset parameter code
498 0491 2      | RETURN ss$normal
499 0492 2      | END;                           ! End of mom$save_node_id

```

	0FFC 00000	.ENTRY	MOMSSAVE_NODE_ID, Save R2,R3,R4,R5,R6,R7,- : 0406
5B 00000000G	00 9E 0002	MOVAB	MOM\$GL_SVD_INDEX, R11
5A 00000000G	8F D0 00009	MOVL	#SVDSGR_PCNO_LAN, R10
59 00000000G	8F D0 00010	MOVL	#SVDSGK_PCNO_LPN, R9
58 00000000G	8F D0 00017	MOVL	#SVDSGK_PCNO_IHO, R8
57 00000000G	00 9E 0001E	MOVAB	MOM\$AB_SERVICE_DATA+9, R7
01 20	AC D1 00025	CMPL	32(NPARSE_BLOCK), #1
	08 12 00029	BNEQ	1\$
	52 D4 00028	CLRL	LENGTH

	53	14	AC 00 0002D	MOVL 20(NPARSE_BLOCK), ADDR	: 0439
			09 11 00031	BRB 2S	: 0436
	53	14	BC 9A 00033	MOVZBL 20(NPARSE_BLOCK), LENGTH	: 0446
			01 C1 00037	ADDL3 #1, 20(NPARSE_BLOCK), ADDR	: 0447
			68 D0 0003C	MOVL MOMSGL_SVD_INDEX, R0	: 0449
			50 D1 0003F	CMPL R0, R8	: 0451
			0C 12 00042	BNEQ 3S	: 0453
			51 58 D0 00044	MOVL R8, NODE_ADDR_SVD	: 0454
			50 00000000G 8F D0 00047	MOVL #SVD\$GK_P[NODE_NAME], NODE_NAME_SVD	: 0454
			20 11 0004E	BRB 5S	: 0449
			59 50 D1 00050	CMPL R0, R9	: 0456
			0C 12 00053	BNEQ 4S	: 0458
			51 59 D0 00055	MOVL R9, NODE_ADDR_SVD	: 0459
			50 00000000G 8F D0 00058	MOVL #SVD\$GK_P[NODE_NAME], NODE_NAME_SVD	: 0449
			0F 11 0005F	BRB 5S	: 0461
			5A 50 D1 00061	CMPL R0, R10	: 0463
			0A 12 00064	BNEQ 5S	: 0464
			51 5A D0 00066	MOVL R10, NODE_ADDR_SVD	: 0471
			50 00000000G 8F D0 00069	MOVL #SVD\$GK_P[NODE_NAME], NODE_NAME_SVD	: 0471
			52 D5 00070	TSTL LENGTH	: 0477
			1E 12 00072	BNEQ 6S	: 0479
	50	51 00000089	8F C4 00074	MULL2 #137, R1	: 0480
		51	57 C1 0007B	ADDL3 R7, R1, R0	: 0485
		60	63 3C 0007F	MOVZWL (ADDR), (R0)	: 0485
		FE A741	01 88 00082	BISB2 #1, MOM\$AB_SERVICE_DATA+7[R1]	: 0493
		00000000V 00	50 DD 00087	PUSHL R0	: 0495
			01 FB 00089	CALLS #1, MOM_FIX_NODE_NUM	: 0496
			17 11 00090	BRB 7S	: 0499
	56	50 00000089	8F C5 00092	MULL3 #137, NODE_NAME_SVD, R6	: 0501
		FF A746	52 90 0009A	MOVB LENGTH, MOM\$AB_SERVICE_DATA+8[R6]	: 0503
		63	52 28 0009F	MOVC3 LENGTH, (ADDR), MOM\$AB_SERVICE_DATA+9[R6]	: 0471
		FE A746	01 88 000A4	BISB2 #1, MOM\$AB_SERVICE_DATA+7[R6]	: 0496
			6B D4 000A9	CLRL MOMSGL_SVD_INDEX	: 0499
		50	01 D0 000AB	MOVL #1, R0	: 0501
			04 000AE	RET	: 0503

: Routine Size: 175 bytes, Routine Base: \$CODE\$ + 013B

```

512      0504 1 XSB'TL 'mom$check_node_entity           Verify a node request'
513      0505 1 GLOBAL ROUTINE mom$check_node_entity =
514      0506 1 ++
515      0507 1 : FUNCTIONAL DESCRIPTION:
516      0508 1 :
517      0509 1 : This is an NPARSE action routine that verifies the requested
518      0510 1 : service request (LOAD/TRIGGER/DUMP) is a node request and
519      0511 1 : not a circuit request. The routine is called whenever a
520      0512 1 : service request containing a service circuit is received.
521      0513 1 :
522      0514 1 : IMPLICIT INPUTS:
523      0515 1 : NPARSE_BLOCK (pointed to by AP) contains the parsed parameter
524      0516 1 : data.
525      0517 1 :
526      0518 1 : MOMSGB_ENTITY_CODE contains the entity code which indicates if a
527      0519 1 : circuit or node request.
528      0520 1 :
529      0521 1 : ROUTINE VALUE:
530      0522 1 : COMPLETION CODE:
531      0523 1 : If request is a node request SUCCESS is returned.
532      0524 1 : Otherwise a parameter not applicable error (NMASC_STS_PNA) will
533      0525 1 : be signalled.
534      0526 1 :
535      0527 1 : SIDE EFFECTS:
536      0528 1 : If error then message is signalled.
537      0529 1 :
538      0530 1 :--
539      0531 2 BEGIN
540      0532 2 :
541      0533 2 $npa_argdef;           ! Define NPARSE block reference
542      0534 2 :
543      0535 2 : Verify that request is not a circuit request (node request).
544      0536 2 : Signal error if circuit request.
545      0537 2 :
546      0538 2 IF .mom$gb_entity_code NEQ mom$sc_node AND
547      0539 2 .mom$gb_entity_code NEQ mom$sc_nodebyname THEN
548      0540 2   mom$error (nmasc_sts_pna,
549      0541 2     .(nparse_block [npa$1_fldptr])<0,16>);
550      0542 2 :
551      0543 2 RETURN SUCCESS
552      0544 1 END:                  ! End MOM$CHECK_NODE_ENTITY routine

```

				.ENTRY	MOM\$CHECK_NODE_ENTITY	Save nothing	: 0505
	50 00000000G	00 9A 00002		MOVZBL	MOMSGB_ENTITY_CODE, R0		: 0538
		13 13 00009		BEQL	1\$: 0539
	01	50 91 00008		CMPB	R0, #1		: 0541
		0E 13 0000E		BEQL	1\$: 0540
	7E	14 BC 3C 00010		MOVZWL	a20(NPARSE_BLOCK), -(SP)		: f
	7E	16 CE 00014		MNEG	#22, -(SP)		
	00 0000000G	02 FB 00017		CALLS	#2, MOM\$ERROR		
	50	01 D0 0001E 1\$:		MOVL	#1, R0		
		04 00021		RET			

MOMPARSE
V04-000

Maintenance Operations NPARSE action routines f 16-Sep-1984 02:06:08
K 7
mom\$check_node_entity Verify a node request 14-Sep-1984 12:44:36

VAX-11 Bliss-32 V4.0-742 Page 19
DISK\$VMSMASTER:[MOM.SRC]MOMPARSE.B32;1 (9)

: Routine Size: 34 bytes. Routine Base: \$CODE\$ + 01EA

MOI
VOI

L 7

Maintenance Operations NPARSE action routines f 16-Sep-1984 02:06:08 VAX-11 Bliss-32 V4.0-742
mom\$check_loop_params Verify LOOP CIRCUIT para 14-Sep-1984 12:44:36 DISK\$VMSMASTER:[MOM.SRC]MOMPARSE.B32;1 (10)

```

554 0545 1 XSBTTL 'mom$check_loop_params'      Verify LOOP CIRCUIT parameters'
555 0546 1 GLOBAL ROUTINE mom$check_loop_params =
556 0547 1 ++
557 0548 1 : FUNCTIONAL DESCRIPTION:
558 0549 1 :
559 0550 1 : This is an NPARSE action routine that verifies the requested
560 0551 1 : LOOP CIRCUIT command does not contain contradictory or missing
561 0552 1 : parameters.
562 0553 1 :
563 0554 1 : IMPLICIT INPUTS:
564 0555 1 :      NPARSE_BLOCK (pointed to by AP) contains the parsed parameter
565 0556 1 :      data.
566 0557 1 :      The Service Data table (SVD)
567 0558 1 :
568 0559 1 : ROUTINE VALUE:
569 0560 1 : COMPLETION CODE:
570 0561 1 :      If request OK, SUCCESS is returned.
571 0562 1 :      Otherwise a parameter missing error (NMASC_STS_PMS) will
572 0563 1 :      be signalled.
573 0564 1 :
574 0565 1 : SIDE EFFECTS:
575 0566 1 :      If error then message is signalled.
576 0567 1 :
577 0568 1 : --
578 0569 2 BEGIN
579 0570 2 :
580 0571 2 $npa_argdef;      ! Define NPARSE block reference
581 0572 2 :
582 0573 2 : If the LOOP CIRCUIT command specifies loop with assist and/or help type,
583 0574 2 : it must be an Ethernet circuit, and therefore a PHYSICAL ADDRESS or NODE
584 0575 2 : parameter must be specified.
585 0576 2 :
586 0577 3 IF (.mom$gl_service_flags [mom$sv_loop_w_assist] OR
587 0578 2 :      .mom$ab_service_data [svd$gk_pcno_lph, svd$sv_msg_param]) AND
588 0579 3 :      NOT (.mom$ab_service_data [svd$gk_pcno_pha, svd$sv_msg_param] OR
589 0580 3 :      .mom$ab_service_data [svd$gk_pcno_lan, svd$sv_msg_param] OR
590 0581 3 :      .mom$ab_service_data [svd$gk_pcno_slna, svd$sv_msg_param]) THEN
591 0582 2 :      mom$error (nmasc_sts_pms,
592 0583 2 :              nmasc_pcno_pha);
593 0584 2 :
594 0585 2 : If the LOOP CIRCUIT command specifies LOOP HELP but no ASSISTANT
595 0586 2 : PHYSICAL ADDRESS or NODE, return an error.
596 0587 2 :
597 0588 2 IF .mom$ab_service_data [svd$gk_pcno_lph, svd$sv_msg_param] AND
598 0589 2 :      NOT .mom$gl_service_flags [mom$sv_loop_w_assist] THEN
599 0590 2 :      mom$error (nmasc_sts_pms,
600 0591 2 :              nmasc_pcno_lpa);
601 0592 2 :
602 0593 2 RETURN success
603 0594 1 END;      ! End MOM$CHECK_LOOP_PARAMS routine

```

07	52 0000000G	00 9E 00009	MOVAB	MOM\$ERROR, R2	:	0577
	63 0000000	03 E0 00010	BBS	#3, MOM\$GL_SERVICE_FLAGS, 1\$:	0578
	32 0000000*	00 E9 00014	BLBC	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_LPH*137>->+7>, 3\$:	
	16 0000000*	00 E8 0001B 1\$:	BLBS	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_PHA*137>->+7>, 2\$:	0579
	0F 0000000*	00 E8 00022	BLBS	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_LAN*137>->+7>, 2\$:	0580
	08 0000000*	00 E8 00029	BLBS	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_SLNA*-137>>+7>, 2\$:	0581
		0A DD 00030	PUSHL	#10	:	0582
	7E	1D CE 00032	MNEG	#29, -(SP)	:	
	62	02 FB 00035	CALLS	#2, MOM\$ERROR	:	
	0E 0000000*	00 E9 00038 2\$:	BLBC	<<MOM\$AB_SERVICE_DATA+<SVD\$GK_PCNO_LPH*137>->+7>, 3\$:	0588
0A	63	03 E0 0003F	BBS	#3, MOM\$GL_SERVICE_FLAGS, 3\$:	0589
	7E 99	8F 9A 00043	MOVZBL	#153, -(SPT)	:	0590
	7E	1D CE 00047	MNEG	#29, -(SP)	:	
	62	02 FB 0004A	CALLS	#2, MOM\$ERROR	:	
	50	01 D0 0004D 3\$:	MOVL	#1, R0	:	0593
		04 00050	RET		:	0594

: Routine Size: 81 bytes. Routine Base: \$CODE\$ + 020C

```

605 0595 1 %SBTTL 'mom$save_mop_msg
606 0596 1 GLOBAL ROUTINE mom$save_mop_msg =
607 0597 1 ++
608 0598 1 FUNCTIONAL DESCRIPTION:
609 0599 1
610 0600 1 This is an NPARSE action routine that is called when certain
611 0601 1 MOP messages are received from the target node. These messages
612 0602 1 must be saved because, if the target does not receive a response
613 0603 1 within a certain time, the target retransmits them. Therefore,
614 0604 1 MOM must be prepared to skip over retransmissions of the same
615 0605 1 message. So, save the message here to do the comparison for
616 0606 1 retransmissions against.
617 0607 1
618 0608 1 IMPLICIT INPUTS:
619 0609 1 NPARSE_BLOCK (pointed to by AP) contains the parsed parameter
620 0610 1 data.
621 0611 1
622 0612 1 ROUTINE VALUE:
623 0613 1 COMPLETION CODE:
624 0614 1
625 0615 1 SIDE EFFECTS:
626 0616 1 The MOP message and a descriptor of it are saved in MOMSAB_MOP_MSG
627 0617 1 and MOMSGQ_MOP_MSG_DSC respectively.
628 0618 1
629 0619 1 --
630 0620 2 BEGIN
631 0621 2
632 0622 2 $npa_argdef; ! Define NPARSE block reference
633 0623 2
634 0624 2 mom$gq_mop_msg_dsc [0] = .mom$ab_nparse_blk [npa$1_msgcnt];
635 0625 2 mom$gq_mop_msg_dsc [1] = mom$ab_mop_msg;
636 0626 2 CHSMOVE (.mom$ab_nparse_blk [npa$1_msgcnt],
637 0627 2 .mom$ab_nparse_blk [npa$1_msgptr],
638 0628 2 mom$ab_mop_msg);
639 0629 2 RETURN success
640 0630 1 END; ! End mom$save_mop_msg routine

```

			007C 00000	.ENTRY	MOMSSAVE_MOP_MSG, Save R2,R3,R4,R5,R6	:	0596
	56 00000000G	00	9E 00002	MOVAB	MOMSAB_MOP_MSG, R6	:	0624
	51 00000000G	00	00 00009	MOVL	MOMSAB_NPARSE_BLK+4, R1	:	0625
00000000G	00	51	00 00010	MOVL	R1, MOMSGQ_MOP_MSG_DSC	:	0627
00000000G	00	66	9E 00017	MOVAB	MOMSAB_MOP_MSG, MOMSGQ_MOP_MSG_DSC+4	:	0626
	50 00000000G	00	00 0001E	MOVL	MOMSAB_NPARSE_BLK+8, R0	:	0629
66	60	51	28 00025	MOV C3	R1, (R0), MOMSAB_MOP_MSG	:	0630
	50	01	00 00029	MOVL	#1, R0	:	
			04 0002C	RET		:	

; Routine Size: 45 bytes, Routine Base: SCODES + 025D

```
: 642 0631 1 %SBTTL 'mom$mop_chk_loop_assist' Check for MOP loop assist request'
: 643 0632 1 GLOBAL ROUTINE mom$mop_chk_loop_assist =
: 644 0633 1 ++
: 645 0634 1 FUNCTIONAL DESCRIPTION:
: 646 0635 1
: 647 0636 1 This is an NPARSE action routine that is called during autoservice
: 648 0637 1 if a MOP messages is received which doesn't contain any of the
: 649 0638 1 recognized MOP function codes. In this case, it could be a
: 650 0639 1 multicast request for loopback assistance on the Ethernet. Check to
: 651 0640 1 make sure the message was sent to the cross company Loopback Assistance
: 652 0641 1 multicast address. If so, return success so the volunteer assistance
: 653 0642 1 will be sent.
: 654 0643 1
: 655 0644 1 IMPLICIT INPUTS:
: 656 0645 1 NPARSE_BLOCK (pointed to by AP) contains the parsed parameter
: 657 0646 1 data.
: 658 0647 1
: 659 0648 1 ROUTINE VALUE:
: 660 0649 1 COMPLETION CODE:
: 661 0650 1 Returns MOMS_SUC if the system sending the MOP message sent it
: 662 0651 1 to the NI multicast loopback assistance address.
: 663 0652 1
: 664 0653 1 --
: 665 0654 2 BEGIN
: 666 0655 2
: 667 0656 2 LOCAL
: 668 0657 2     status;
: 669 0658 2
: 670 0659 2 BIND
: 671 0660 2     NI_loop_assis_mult = UPLIT (XX'000000CF', WORD (XX'0000'));
: 672 0661 2
: 673 0662 2
: 674 0663 2 ! Check to make sure the MOP message was sent to the multicast loopback
: 675 0664 2 ! assist address. The destination address of the message was saved
: 676 0665 2
: 677 0666 2 status = mom$badmopfct;
: 678 0667 2 IF .mom$gl_service_flags [mom$sv_ni_circ] THEN
: 679 0668 3     BEGIN
: 680 0669 3     IF CH$EQQL (mom$ki_ni_addr_length, ni_loop_assis_mult,
: 681 0670 3             mom$ki_ni_addr_length,
: 682 0671 3             mom$15_service_data [svd$gk_pcno_sda, svd$1t_string]) THEN
: 683 0672 3     status = success;
: 684 0673 2     END;
: 685 0674 2 RETURN .status;
: 686 0675 1 END;                                ! End mom$mop_chk_loop_assist routine
```

.PSECT \$PLITS,NOWRT,NOEXE,2
000000CF 00000 P.AAA: .LONG 207
0000 00004 .WORD 0

NI_LOOP_ASSIS_MULT= P.AAA

.PSECT \$CODE\$,NOWRT,2

00000000*	11 00000000G	54 00000000G	00 00000000	001C 00000	.ENTRY	MOMSMOP_CHK_LOOP_ASSIST, Save R2,R3,R4	: 0632
	30 00000000*	00	00	8F D0 00002	MOVL	#MOMS_BADMOPFCT_STATUS	: 0666
				01 E1 00009	BBC	#1, M0M\$GL SERVICE FLAGS, 1\$: 0667
				06 29 00011	CMPC3	#6, NI LOOP ASSIS MULT <- <M0M\$AB_SERVICE_DATA+<\$VD\$GK_PCNO_SDA*137>- +9>	: 0671
					BNEQ	1\$: 0672
		54	50	03 12 00010	MOVL	#1, STATUS	: 0674
				01 D0 0001F	MOVL	STATUS, R0	: 0675
				54 D0 00022 1\$:	RET		
				04 00025			

: Routine Size: 38 bytes, Routine Base: \$CODE\$ + 028A

```
0688 0676 1 %SBTTL 'mom$save_load_file_id Save load file specification'
0689 0677 1 GLOBAL ROUTINE mom$save_load_file_id =
0690 0678 1 ++
0691 0679 1 FUNCTIONAL DESCRIPTION:
0692 0680 1
0693 0681 1 This is an NPARSE action routine MOM calls if it receives a
0694 0682 1 MOP program load request which contains string in the software
0695 0683 1 id field of the message. Append the logical name MOM$LOAD
0696 0684 1 to the string. It will be translated by RMS when the load
0697 0685 1 file is opened. The logical name is used as security to make
0698 0686 1 sure that only files in one directory can be loaded.
0699 0687 1
0700 0688 1
0701 0689 1 IMPLICIT INPUTS:
0702 0690 1 NPARSE_BLOCK (pointed to by AP) contains the parsed parameter
0703 0691 1 data.
0704 0692 1
0705 0693 1 ROUTINE VALUE:
0706 0694 1 COMPLETION CODE:
0707 0695 1
0708 0696 1 --
0709 0697 2 BEGIN
0710 0698 2 $npa_argdef; ! Define NPARSE block reference
0711 0699 2
0712 0700 2 LOCAL
0713 0701 2   file_svd,
0714 0702 2   len
0715 0703 2   MOP_ptr,
0716 0704 2   svd_ptr;
0717 0705 2
0718 0706 2
0719 0707 2 The software type field precedes the software ID in the MOP message.
0720 0708 2 This field determines which load file (secondary, tertiary, or operating
0721 0709 2 system) to load. Put the load file id in the correct load file entry
0722 0710 2 of the Service Data Table (SVD).
0723 0711 2
0724 0712 2 file_svd =
0725 0713 3   ?SELECTONEU .mom$ab_service_data [svd$gk_pcno_sty, svd$l_param] OF
0726 0714 3     SET
0727 0715 3     [nma$C_soft_terl]: svd$gk_pcno_tlo; ! Tertiary loader
0728 0716 3     [nma$C_soft_osys]: svd$gk_pcno_loa; ! Operating system
0729 0717 3     [OTHERWISE]: svd$gk_pcno_slo; ! Secondary loader
0730 0718 2   TES;
0731 0719 2
0732 0720 2 Concatenate the logical name, MOM$LOAD, with the file specification
0733 0721 2 in the software ID field of the MOP message.
0734 0722 2
0735 0723 2 len = %CHARCOUNT ('MOM$LOAD:');
0736 0724 2 svd_ptr = mom$ab_service_data [.file_svd, svd$st_string];
0737 0725 2 svd_ptr = CHSMOVE (.len, UPLIT BYTE 'MOM$LOAD:', .svd_ptr);
0738 0726 2
0739 0727 2 Save the software id in the Service Data Table.
0740 0728 2
0741 0729 2 len = .len + .nparse_block [npa$l_fldcnt] - 1;
0742 0730 2 MOP_ptr = .nparse_block [npa$l_fldptr];
0743 0731 2 CHSMOVE (.len, (.MOP_ptr + 1), .svd_ptr);
0744 0732 2 mom$ab_service_data [.file_svd, svd$b_string_len] = .len;
```

```
:
: 745 0733 2 mom$ab_service_data [.file_svd, svd$sv_msg_param] = true;
: 746 0734 2
: 747 0735 2 RETURN success;
: 748 0736 1 END;           ! End mom$save_load_file_id routine
```

```
.PSECT $PLIT$,NOWRT,NOEXE,2
3A 44 41 4F 4C 24 4D 4F 4D 00006 P.AAB: .ASCII \MOM$LOAD:\;
```

			.PSECT \$CODE\$,NOWRT,2	
			.ENTRY MOMSSAVE_LOAD_FILE_ID, Save R2,R3,R4,R5,R6,-; 0677	
			R7,R8	
			MOVAB MOMSAB_SERVICE_DATA+9, R8	
			MOVL <<MOMSAB_SERVICE_DATA+<SVDSGK_PCNO_STY*137>->+9>, R0	0713
			CMPL R0, #1	0715
			BNEQ 1\$	
			MOVL #SVDSGK_PCNO_TLO, FILE_SVD	
			BRB 3\$	
			CMPL R0, #2	0716
			BNEQ 2\$	
			MOVL #SVDSGK_PCNO_LOA, FILE_SVD	
			BRB 3\$	
			MOVL #SVDSGK_PCNO_SLO, FILE_SVD	0717
			MOVL #9, LEN	0723
			MULL3 #137, FILE_SVD, R7	0724
			ADDL3 R8, R7, SVDPTR	
			MOVC3 LEN, P.AAB, -(SVDPTR)	0725
			ADDL3 16(NPARSE_BLOCK), LEN, R0	0729
			MOVAB -1(R0), LEN	
			MOVL 20(NPARSE_BLOCK), MOP_PTR	0730
			MOVC3 LEN, 1(MOP_PTR), (SVDPTR)	0731
			MOVB LEN, MOMSAB_SERVICE_DATA+8[R7]	0732
			BISB2 #1, MOMSAB_SERVICE_DATA+7[R7]	0733
			MOVL #1, R0	0735
			RET	0736

```
: Routine Size: 106 bytes, Routine Base: $CODE$ + 02B0
```

```
0737 1 %SBTTL 'MOM_FIX_NODE_NUM' Fix node address parameter (action routine)'
0738 1 ROUTINE MOM_FIX_NODE_NUM (NODE_ADDR) : NOVALUE =
0739 1
0740 1 /**
0741 1 FUNCTIONAL DESCRIPTION:
0742 1
0743 1 This is an parsing action routine that checks the node address. If
0744 1 the area number is 0 it can be one of two cases:
0745 1 The NCP is a Phase IV NCP, and user did not specify an area
0746 1 number in the NCP command. In this case, assume the user
0747 1 means area 1 (since 0 is an invalid area number).
0748 1
0749 1 the NCP is a Phase III NCP and therefore doesn't understand
0750 1 area numbers. In this case, assume the user means the
0751 1 executor node's area.
0752 1
0753 1 FORMAL PARAMETERS:
0754 1 NODE_ADDR Address of Node address to fix.
0755 1
0756 1 IMPLICIT INPUTS:
0757 1 None
0758 1
0759 1 IMPLICIT OUTPUTS:
0760 1 None
0761 1
0762 1 --
0763 1
0764 2 BEGIN
0765 2
0766 2 MAP
0767 2 node_addr : REF BBLOCK [2];
0768 2
0769 2 LOCAL
0770 2 exec_addr : BBLOCK [2];
0771 2
0772 2
0773 2 If the node address is 0, then it's the executor, so leave it that way.
0774 2 If the area number of the address is 0, then change it.
0775 2
0776 2 IF .node_addr [nma$v_addr] NEQ 0 AND
0777 2 .node_addr [nma$v_area] EQL 0 THEN
0778 3 BEGIN
0779 3
0780 3 If NCP is a Phase III NCP, use area 0 for the volatile database.
0781 3 NETACP will assume the executor's area number. For permanent database,
0782 3 get the exec address from the permanent database and use it's area number.
0783 3
0784 3 IF CHSRCHAR (mom$ab_ncp_version) LEQ 3 then
0785 3 node_addr [nma$v_area] = 0
0786 3 ELSE
0787 3
0788 3 If NCP is a Phase IV NCP, use area 1.
0789 3
0790 3 node_addr [nma$v_area] = 1;
0791 2 END;
0792 2
0793 2 RETURN
```

: 807

0794 1 END;

! End of MOM_FIX_NODE_NUM

0000 00000 MOM_FIX_NODE_NUM:									
						WORD	Save nothing		0738
03FF	50	04	AC	D0	00002	MOVL	NODE_ADDR, R0		0776
	8F		60	B3	00006	BITW	(R0), #1023		
			1B	13	0000B	BEQL	2\$		
FC	8F	01	A0	93	0000D	BITB	1(R0), #252		0777
			14	12	00012	BNEQ	2\$		
		03	00000000G	00	91	CMPB	MOM\$AB_NCP_VERSION, #3		0784
				06	1A	BGTRU	1\$		
01	A0	FC	8F	8A	0001B	BICB2	#252, 1(R0)		0785
				04	00022	RET			
60	06	0A	01	F0	00023	INSV	#1, #10, #6, (R0)		0790
				04	00028	2\$:	RET		0794

: Routine Size: 41 bytes, Routine Base: SCODE\$ + 031A

```

: 809 0795 1 %SBTTL 'mom$parse_error Build and signal error (action routine)'
: 810 0796 1 GLOBAL ROUTINE mom$parse_error : NOVALUE =
: 811 0797 1
: 812 0798 1 ++
: 813 0799 1 FUNCTIONAL DESCRIPTION:
: 814 0800 1 This NPARSE action routine is called if an error is found when parsing
: 815 0801 1 the parameters of a NICE command message. It signals the error.
: 816 0802 1
: 817 0803 1 FORMAL PARAMETERS:
: 818 0804 1 NONE
: 819 0805 1
: 820 0806 1 IMPLICIT INPUTS:
: 821 0807 1 NPARSE argument block.
: 822 0808 1 NPASL_PARAM contains the error code.
: 823 0809 1 NPASL_FLDPTR points to the parameter in the message.
: 824 0810 1
: 825 0811 1 SIDE EFFECTS:
: 826 0812 1 Error message is signalled.
: 827 0813 1
: 828 0814 1 --
: 829 0815 1
: 830 0816 2 BEGIN
: 831 0817 2
: 832 0818 2 $npa_argdef; ! Define NPARSE block reference
: 833 0819 2
: 834 0820 2 LOCAL
: 835 0821 2 err_code; ! Error code
: 836 0822 2 err_detail; ! Error detail
: 837 0823 2
: 838 0824 2 err_code = .nparse_block [npa$1_param]; ! Get error code
: 839 0825 2
: 840 0826 2 Check for parameters to move in addition to error status.
: 841 0827 2
: 842 0828 3 err_detail = (
: 843 0829 3   SELECTONEU .err_code OF
: 844 0830 3     SET
: 845 0831 3     [nma$c_sts_pty,
: 846 0832 3     nma$c_sts_pva,
: 847 0833 3     nma$c_sts_pna];
: 848 0834 3     .7.nparse_block [npa$1_msgptr] - 2)<0,16,0>; ! Get detail code
: 849 0835 3
: 850 0836 3 [OTHERWISE]:
: 851 0837 3   -1;
: 852 0838 3
: 853 0839 2   TES);
: 854 0840 2
: 855 0841 2 mom$error (.err_code, .err_detail); ! Signal error message
: 856 0842 2
: 857 0843 1 END; ! End of MOMPARSE_ERROR

```

FFFFFEA	51	20	0000 00000	.ENTRY MOMPARSE_ERROR, Save nothing	: 0796
	8F		AC D0 00002	MOVL 32(NPARSE_BLOCK), ERR_CODE	: 0824
			51 D1 00006	CMPL ERR_CODE, #-22	: 0831

FFFFFFF0	8F	12 13 00000	BEQL	1\$:	
		51 D1 0000F	CMPL	ERR_CODE, #-16		
FFFFFFF4	8F	09 13 00016	BEQL	1\$		
		51 D1 00018	CMPL	ERR_CODE, #-6		
		0A 12 0001F	BNEQ	2\$		
50	08	AC D0 00021	1\$:	MOVL	8(NPARSE_BLOCK), R0	
50	FE	A0 3C 00025	MOVZWL	-2(R0), ERR_DETAIL	0834	
		03 11 00029	BRB	3\$		
50		01 CE 0002B	2\$:	MNEG	#1, ERR_DETAIL	0837
		50 DD 0002E	3\$:	PUSHL	ERR_DETAIL	0841
		51 DD 00030	PUSHL	ERR_CODE		
00000000G	00	02 FB 00032	CALLS	#2, MOM\$ERROR		
		04 00039	RET		0843	

: Routine Size: 58 bytes, Routine Base: \$CODES + 0343

```

: 859      0844 1 %SBTTL 'mom$prsmoperr MOP parameter parsing error'
: 860      0845 1 GLOBAL ROUTINE mom$prsmoperr =
: 861      0846 1
: 862      0847 1 ++
: 863      0848 1 FUNCTIONAL DESCRIPTION:
: 864      0849 1
: 865      0850 1 This routine sets up response message information for errors
: 866      0851 1 encountered in parsing MOP messages.
: 867      0852 1
: 868      0853 1 FORMAL PARAMETERS:
: 869      0854 1
: 870      0855 1 NONE
: 871      0856 1
: 872      0857 1 IMPLICIT INPUTS:
: 873      0858 1
: 874      0859 1 The NPARSE argument block (NPASL_PARAM) contains the code for
: 875      0860 1 the optional text message to be signalled.
: 876      0861 1
: 877      0862 1 IMPLICIT OUTPUTS:
: 878      0863 1
: 879      0864 1 MOM$AB_MSGBLOCK contains the response message information.
: 880      0865 1
: 881      0866 1 --
: 882      0867 1
: 883      0868 2 BEGIN
: 884      0869 2
: 885      0870 2 $npa_argdef:
: 886      0871 2
: 887      0872 2
: 888      0873 2 Set up MOP protocol error with optional text message.
: 889      0874 2
: 890      0875 2 mom$ab_msgblock [msb$1_flags] = msb$1_msg_fld;
: 891      0876 2 mom$ab_msgblock [msb$1_code] = nma$1_sts_lpr;
: 892      0877 2 mom$ab_msgblock [msb$1_text] = .nparse_block [npasl_param];
: 893      0878 2
: 894      0879 2 RETURN success
: 895      0880 2
: 896      0881 1 END;           ! End of mom$prsmoperr

```

0004 00000	.ENTRY	MOMSPRSMOPERR, Save R2	: 0845
52 0000000G 00 9E 00002	MOVAB	MOM\$AB_MSGBLOCK, R2	: 0875
04 62 04 D0 00009	MOVL	#4, MOM\$AB_MSGBLOCK	: 0876
0C A2 11 8E 0000C	MNEG8	#17, MOM\$AB_MSGBLOCK+4	: 0877
20 AC 00 00010	MOVL	32(NPARSE_BLOCK), MOM\$AB_MSGBLOCK+12	: 0879
50 01 00 00015	MOVL	#1, R0	: 0881
04 00018	RET		

: Routine Size: 25 bytes, Routine Base: \$CODE\$ + 037D

```

: 897      0882 1
: 898      0883 1
: 899      0884 1

```

MOPPARSE
V04-000

Maintenance Operations NPARSE action routines f K 8
MOM\$PRSMOPERR MOP parameter parsing error 16-Sep-1984 02:06:08
14-Sep-1984 12:44:36

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[MOM.SRC]MOPPARSE.B32;1 (16)

Page 32
MO
VO

: 900 0885 1 END
: 901 0886 1
: 902 0887 0 ELUDOM

! End of module

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	918 NOVEC,NOWRT; RD	EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$SPLIT\$	15 NOVEC,NOWRT; RD	;NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Symbols -----	Pages	Processing	
	Total	Loaded	Mapped	Time
-\$255\$DUA28:[MOM.OBJ]MOMLIB.L32;1	194	33	17	21 00:00.1
-\$255\$DUA28:[SHRLIB]NMALIBRY.L32;1	887	11	1	47 00:00.2
-\$255\$DUA28:[SHRLIB]NET.L32;1	1279	0	0	63 00:00.3
-\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	1	0	581 00:03.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:MOPPARSE/OBJ=OBJ\$:MOPPARSE MSRC\$:MOPPARSE/UPDATE=(LNH\$:MOPPARSE)

: Size: 918 code + 15 data bytes

: Run Time: 00:21.5

: Elapsed Time: 00:48.7

: Lines/CPU Min: 2471

: Lexemes/CPU-Min: 12398

: Memory Used: 107 pages

: Compilation Complete

0238 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

